EQLC-P01-006

CLIENT LOAD DISTRIBUTION

Inventors:
G. Paul Koning
Peter C. Hayden
Paula Long
Daniel Suman

Reference to Related Application

This application claims priority to US Provisional Application USSN 60/441,810 filed 1/21/2003 and naming G. Paul Koning, among others, as an inventor, the contents of which are incorporated by reference.

Background of the Invention

The invention relates to data storage and in particular to data block storage services that store data blocks across a plurality of servers.

The client/server architecture has been one of the more successful innovations in information technology. The client/server architecture allows a plurality of clients to access services and resources maintained and/or controlled by a server. The server listens for, and responds to, requests from the clients and in response to the request determines whether or not the request can be satisfied. The server responds to the client as appropriate. A typical example of a client/server system is where a server is set up to store data files and a number of different clients can communicate with the server for the purpose of requesting the server to grant access to different ones of the data files

maintained by the file server. If a data file is available and a client is authorized to access that data file, the server can deliver the requested data file to the server and thereby satisfy the client's request.

Although the clientserver architecture has worked remarkably well it does have some drawbacks. In particular, the clientserver environment is somewhat dynamic. For example, the number of clients contacting a server and the number of requests being made by individual clients can vary significantly over time. As such, a server responding to client requests may find itself inundated with a volume of requests that is impossible or nearly impossible to satisfy. To address this problem, network administrators often make sure that the server includes sufficient data processing resources to respond to anticipated peak levels of client requests. Thus, for example, the network administrator may make sure that the server comprises a sufficient number of central processing units (CPUs) with sufficient memory and storage space to handle the volume of client traffic that may arrive.

Even with a studied provisioning of resources, variations in client load can still burden a server system. For example, even if sufficient hardware resources are provided in the server system, it may be the case that client requests focus on a particular resource maintained by the server and supported by only a portion of the available resources. Thus, continuing with our above example, it is not uncommon that client requests overwhelmingly focus on a small portion of the data files maintained by the file server.

Accordingly, even though the file server may have sufficient hardware resources to respond to a certain volume of client requests, if these requests are focused on a particular resource, such as a particular data file, most of the file server resources will remain idle while those resources that support the data file being targeted by the plurality of clients are over burdened.

To address this problem, network engineers have developed load balancing systems that act as a gateway to client requests and distribute client requests across the available resources for the purpose of distributing client load. To this end, the gateway system may distribute client requests in a round-robin fashion that evenly distributes requests across the available server resources. In other practices, the network administrator sets up a replication system that can identify when a particular resource is the subject of a flurry of client requests and duplicates the targeted resource so that more of the server resources are employed in supporting client requests for that resource.

Although the above techniques may work well with certain server architectures, they each require that a central transaction point be disposed between the clients and the server. As such, this central transaction point may act as a bottle neck that slows the servers response to client requests. Accordingly, there is a need in the art for a method for distributing client load across a server system while at the same time providing suitable response times for these incoming client requests.

Summary of the Invention

The systems and methods described herein, include systems for managing requests for a plurality of clients for access to a set of resources. In one embodiment, the systems comprise a plurality of servers wherein the set of resources is partitioned across this plurality of servers. Each server has a load monitor process that is capable of communicating with the other load monitor processes for generating a measure of the client load on the server system and the client load on each of the respective servers.

Accordingly, in one embodiment, the systems comprise a server system having a plurality of servers, each of which has a load monitor process that is capable of coordinating with other load monitor processes executing on other servers to generate a system-wide view of the client load being handled by the server system and by individual respective servers.

Optionally, the systems may further comprise a client distribution process that is responsive to the measured system load and is capable of repartitioning the set of client connections among the server systems to thereby redistribute the client load.

Accordingly, it will be understood that the systems and methods described herein include client distribution systems that may work with a partitioned service, wherein the partitioned service is supported by a plurality of equivalent servers each of which is responsible for a portion of the service that has been partitioned across the equivalent servers. In one embodiment each equivalent server is capable of monitoring the relative

load that each of the clients that server is communicating with is placing on the system and on that particular server. Accordingly, each equivalent server is capable of determining when a particular client would present a relative burden to service. However, for a partitioned service each client is to communicate with that equivalent server that is responsible for the resource of interest of the client. Accordingly, in one embodiment, the systems and methods described herein redistributed client load by, in part, redistributing resources across the plurality of servers.

In another embodiment, the systems and methods described herein include storage area network systems that may be employed for providing storage resources for an enterprise. The storage area network (SAN) of the invention comprises a plurality of servers and/or network devices that operate on the storage area network. At least a portion of the servers and network devices operating on the storage area network include a load monitor process that monitors the client load being placed on the respective server or network device. The load monitor process is further capable of communicating with other load monitor processes operating on the storage area network. The load monitor process on the server is capable of generating a system-wide load analysis that indicate the client load being placed on the storage area network. Additionally, the load monitor process is capable of generating an analysis of the client load being placed on that respective server and/or network device. Based on the client load information observed by the load monitor process, the storage area network is capable of redistributing client

load to achieve greater responsiveness to client requests. In one embodiment, the storage area network is capable of moving the client connections supported by the system for the purpose of redistributing client load across the storage area network.

Further features and advantages of the present invention will be apparent from the following description of preferred embodiments and from the claims.

Brief Description of the Drawings

The following figures depict certain illustrative embodiments of the invention in which like reference numerals refer to like elements. These depicted embodiments are to be understood as illustrative of the invention and not as limiting in any way.

FIG. 1 is a schematic diagram of a client-server architecture with servers organized in server groups;

FIG. 2 is a schematic diagram of the server groups as seen by a client;

FIG. 3 shows details of the information flow between the client and the servers of a group;

FIG. 4 is a process flow diagram for retrieving resources in a partitioned resource environment;

FIG. 5 depicts in more detail and as a functional block diagram one embodiment of a system according to the invention; and

FIG. 6 depicts one example of a routing table suitable for use with the systems

and methods described herein.

Detailed Description of Certain Illustrated Embodiments

The systems and methods described herein include systems for organizing and managing resources that have been distributed over a plurality of servers on a data network. More particularly, the systems and methods described herein include systems and methods for providing more efficient operation of a partitioned service. The type of service can vary, however for purpose of illustration the invention will be described with reference to systems and methods for managing the allocation of data blocks across a partitioned volume of storage. It will be understood by those of skill in the art that the other applications and services may include, although are not limited to, distributed file systems, systems for supporting application service providers and other applications. Moreover, it will be understood by those of ordinary skill in the art that the systems and methods described herein are merely exemplary of the kinds of systems and methods that may be achieved through the invention and that these exemplary embodiments may be modified, supplemented and amended as appropriate for the application at hand.

Referring first to FIG. 1 one embodiment of a system according to the invention is depicted. As show in FIG. 1, one or several clients 12 are connected, for example via a network 14, such as the Internet, an intranet, a WAN or LAN, or by direct connection, to servers 161, 162, and 163 that are part of a server group 16.

The client 12 can be any suitable computer system such as a PC workstation, a

handheld computing device, a wireless communication device, or any other such device, equipped with a network client program capable of accessing and interacting with the server 16 to exchange information with the server 16. Optionally, the client 12 and the server 16 rely on an unsecured communication path for accessing services at the remote server 16. To add security to such a communication path, the client 12 and the server 16 may employ a security system, such as any of the conventional security systems that have been developed to provide to the remote user a secured channel for transmitting data over a network. One such system is the Netscape secured socket layer (SSL) security mechanism that provides to a remote user a trusted path between a conventional web browser program and a web server.

FIG. 1 further depicts that the client 12 may communicate with a plurality of servers, 161, 162 and 163. The servers 161, 162 and 163 employed by the system 10 may be conventional, commercially available server hardware platforms, such as a Sun Sparc™ systems running a version of the Unix operating system. However any suitable data processing platform may be employed. Moreover, it will be understood that one or more of the servers 161, 162 or 163 may comprise a network device, such as a tape library, or other device, that is networked with the other servers and clients through network 14.

Each server 161, 162 and 163 may include software components for carrying out the operation and the transactions described herein, and the software architecture of the

servers 161, 162 and 163 may vary according to the application. In certain embodiments, the servers 161, 162 and 163 may employ a software architecture that builds certain of the processes described below into the server's operating system, into device drivers, into application level programs, or into a software process that operates on a peripheral device, such as a tape library, a RAID storage system or some other device. In any case, it will be understood by those of ordinary skill in the art, that the systems and methods described herein may be realized through many different embodiments, and practices, and that the particular embodiment and practice employed will vary as a function of the application of interest and all these embodiments and practices fall within the scope hereof.

In operation, the clients 12 will have need of the resources partitioned across the server group 16. Accordingly, each of the clients 12 will send requests to the server group 16. The clients typically act independently, and as such, the client load placed on the server group 16 will vary over time. In a typical operation, a client 12 will contact one of the servers, for example server 161, in the group 16 to access a resource, such as a data block, page, file, database, application, or other resource. The contacted server 161 itself may not hold or have control over the requested resource. However, in a preferred embodiment, the server group 16 is configured to make all the partitioned resources available to the client 12 regardless of the server that initially receives the request. For illustration, the diagram shows two resources, one resource 18 that is

partitioned over all three servers, servers 161, 162, 163, and another resource 17 that is partitioned over two of the three servers. In the exemplary application of the system 10 being a block data storage system, each resource 18 and 17 may represent a partitioned block data volume.

In the embodiment of Fig. 1, the server group 16 therefore provides a block data storage service that may operate as a storage area network (SAN) comprised of a plurality of equivalent servers, servers 161, 162 and 163. Each of the servers 161, 162 and 163 may support one or more portions of the partitioned block data volumes 18 and 17. In the depicted system 10, there are two data volumes and three servers, however there is no specific limit on the number of servers. Similarly, there is no specific limit on the number of resources or data volumes. Moreover, each data volume may be contained entirely on a single server, or it may be partitioned over several servers, either all of the servers in the server group, or a subset of the server group. In practice, there may of course be limits due to implementation considerations, for example the amount of memory available in the servers 161, 162 and 163 or the computational limitations of the servers 161, 162 and 163. Moreover, the grouping itself, i.e., deciding which servers will comprise a group, may in one practice comprise an administrative decision. In a typical scenario, a group might at first contain only a few servers, perhaps only one. The system administrator would add servers to a group as needed to obtain the level of service required. Increasing servers creates more space (memory, disk storage) for resources that are stored, more CPU processing capacity to act on the client requests,

and more network capacity (network interfaces) to carry the requests and responses from and to the clients. It will be appreciated by those of skill in the art that the systems described herein are readily scaled to address increased client demands by adding additional servers into the group 16. However, as client load varies, the system 10 as described below can redistribute client load to take better advantage of the available resources in server group 16. To this end, the system 10 in one embodiment, comprises a plurality of equivalent servers. Each equivalent server supports a portion of the resources partitioned over the server group 16. As client requests are delivered to the equivalent servers, the equivalent servers coordinate among themselves to generate a measure of system load and to generate a measure of the client load of each of the equivalent servers. In a preferred practice, this coordinating is done in a manner that is transparent to the clients 12, so that the clients 12 see only requests and responses traveling between the clients 12 and server group 16.

Referring now to FIG. 2, a client 12 connecting to a server 161 (FIG. 1) will see the server group 16 as if the group were a single server having multiple IP addresses. The client 12 is not aware that the server group 16 is constructed out of a potentially large number of servers 161, 162, 163, nor is it aware of the partitioning of the block data volumes 17, 18 over the several servers 161, 162, 163. As a result, the number of servers and the manner in which resources are partitioned among the servers may be changed without affecting the network environment seen by the client 12.

Referring now to FIG. 3, in the partitioned server group 16, any volume may be spread over any number of servers within the group 16. As seen in FIGS. 1 and 2, one volume 17 (Resource 1) may be spread over servers 162, 163, whereas another volume 18 (Resource 2) may be spread over servers 161, 162, 163. Advantageously, the respective volumes are arranged in fixed-size groups of blocks, also referred to as "pages", wherein an exemplary page contains 8192 blocks. Other suitable page sizes may be employed. In an exemplary embodiment, each server in the group 16 contains a routing table 165 for each volume, with the routing table 165 identifying the server on which a specific page of a specific volume can be found. For example, when the server 161 receives a request from a client 12 for volume 3, block 93847, the server 161 calculates the page number (page 11 in this example for the page size of 8192) and looks up in the routing table 165 the location or number of the server that contains page 11. If server 163 contains page 11, the request is forwarded to server 163, which reads the data and returns the data to the server 161. Server 161 then send the requested data to the client 12. In other words, the response is always returned to the client 12 via the same server 161 that received the request from the client 12.

It is transparent to the client 12 to which server 161, 162, 163 he is connected. Instead, the client only sees the servers in the server group 16 and requests the resources of the server group 16. It should be noted here that the routing of client requests is done separately for each request. This allows portions of the resource to exist at different

servers. It also allows resources, or portions thereof, to be moved while the client is connected to the server group 16 – if that is done, the routing tables 165 are updated as necessary and subsequent client requests will be forwarded to the server now responsible for handling that request. At least within a resource 17 or 18, the routing tables 165 are identical. The described invention is different from a "redirect" mechanism, wherein a server determines that it is unable to handle requests from a client, and redirects the client to the server that can do so. The client then establishes a new connection to another server. Since establishing a connection is relatively inefficient, the redirect mechanism is ill suited for handling frequent requests.

FIG. 4 depicts an exemplary request handling process 40 for handling client requests in a partitioned server environment. The request handling process 40 begins at 41 by receiving a request for a resource, such as a file or blocks of a file, at 42. The request handling process 40 checks, in operation 43, if the requested resource is present at the initial server that received the request from the client 12 examines the routing table, in operation 43, to determine at which server the requested resource is located. If the requested resource is present at the initial server, the initial server returns the requested resource to the client 12, at 48, and the process 40 terminates at 49. Conversely, if the requested resource is not present at the initial server, the server will consult a routing table, operation 44, use the data from the routing table to determine which server actually holds the resource requested by the client, operation 45. The request is then

forwarded to the server that holds the requested resource, operation 46, which returns the requested resource to the initial server, operation 48. The process 40 then goes to 48 as before, to have the initial server forward the requested resource to the client 12, and the process 40 terminates, at 49.

The resources spread over the several servers can be directories, individual files within a directory, or even blocks within a file. Other partitioned services could be contemplated. For example, it may be possible to partition a database in an analogous fashion or to provide a distributed file system, or a distributed or partitioned server that supports applications being delivered over the Internet. In general, the approach can be applied to any service where a client request can be interpreted as a request for a piece of the total resource, and operations on the pieces do not require global coordination among all the pieces.

Turning now to FIG. 5, one particular embodiment of a block data service system 10 is depicted. Specifically, FIG. 5 depicts the system 10 wherein the client 12 communicates with the server group 16. The server group 16 includes three servers, server 161, 162 and 163. Each server includes a routing table depicted as routing tables 20A, 20B and 20C. In addition to the routing tables, each of the equivalent servers 161, 162 and 163 are shown in FIG. 5 as including a load monitor process, 22A, 22B and 22C respectively.

As shown in FIG. 5, each of the equivalent servers 161, 162 and 163 may include a

routing table 20A, 20B and 20C respectively. As shown in FIG. 5, each of the routing tables 20A, 20B and 20C are capable of communicating with each other for the purposes of sharing information. As described above, the routing tables can track which of the individual equivalent servers is responsible for a particular resource maintained by the server group 16. In the embodiment shown in FIG. 5 the server group 16 may be a SAN, or part of a SAN, wherein each of the equivalent servers 161, 162 and 163 has an individual IP address that may be employed by a client 12 for accessing that particular equivalent server on the SAN. As further described above, each of the equivalent servers 161, 162 and 163 is capable of providing the same response to the same request from a client 12. To that end, the routing tables of the individual equivalent 161, 162 and 163 coordinate with each other to provide a global database of the different resources, and this exemplary embodiments data blocks, pages or other organizations of data blocks, and the individual equivalent servers that are responsible for those respective data blocks, pages, files or other storage organization.

FIG. 6 depicts an example routing table. Each routing table in the server group 16, such as table 20A, includes an identifier (Server ID) for each of the equivalent servers 161, 162 and 163 that support the partitioned data block storage service. Additionally, each of the routing tables includes a table that identifies those data blocks pages associated with each of the respective equivalent servers. In the embodiment depicted by FIG. 6, the equivalent servers support two partitioned volumes. A first one of the

volumes, Volume 18, is distributed or partitioned across all three equivalent servers 161, 162 and 163. The second partitioned volume, Volume 17, is partitioned across two of the equivalent servers, servers 162 and 163 respectively.

The routing tables may be employed by the system 10 to balance client load across the available servers.

The load monitor processes 22A, 22B and 22C each observe the request patterns arriving at their respective equivalent servers to determine to determine whether patterns or requests from clients 12 are being forwarded to the SAN and whether these patterns can be served more efficiently or reliably by a different arrangement of client connections to the several servers. In one embodiment, the load monitor processes 22A, 22B and 22C monitor client requests coming to their respective equivalent servers. In one embodiment, the load monitor processes each build a table representative of the different requests that have been seen by the individual request monitor processes. Each of the load monitor processes 22A, 22B and 22C are capable of communicating between themselves for the purpose of building a global database of requests seen by each of the equivalent servers. Accordingly, in this embodiment each of the load monitor processes is capable of integrating request data from each of the equivalent servers 161, 162 and 163 in generating a global request database representative of the request traffic seen by the entire block data storage system 16. In one embodiment, this global request database is made available to the client distribution processes 30A, 30B and 30C for their use in

determining whether a more efficient or reliable arrangement of client connections is available.

FIGURE 5 illustrates pictorially that the server group 16 may be capable of redistributing client load by having client 12C, which was originally communicating with server 161, redistributed to server 162. To this end, FIGURE 5 depicts an initial condition wherein the server 161 is communicating with clients 12A, 12B, and 12C. This is depicted by the bidirectional arrows coupling the server 161 to the respective clients 12A, 12B, and 12C. As further shown in FIGURE 5, in an initial condition, clients 12D and 12E are communicating with server 163 and no client (during the initial condition) is communicating with server 162. Accordingly, during this initial condition, server 161 is supporting requests from three clients, clients 12A, 12B, and 12C. Server 162 is not servicing or responding to requests from any of the clients.

Accordingly, in this initial condition the server group 16 may determine that server 161 is overly burdened or asset constrained. This determination may result from an analysis that server 161 is overly utilized given the assets it has available. For example, it could be that the server 161 has limited memory and that the requests being generated by clients 12A, 12B, and 12C have overburdened the memory assets available to server 161. Thus, server 161 may be responding to client requests at a level of performance that is below an acceptable limit. Alternatively, it may be determined that server 161, although performing and responding to client requests at an acceptable level, is overly burdened with respect to the client load (or bandwidth) being carried by server 162. Accordingly, the client distribution process 30 of the server group 16 may make a determination that overall efficiency may be improved by redistributing client load from its initial condition to one wherein server 162 services requests from client 12C. Considerations that drive the load balancing decision may vary and some examples are

the desire to reduce routing: for example if one server is the destination of a significantly larger fraction of requests than the others on which portions of the resource (e.g., volume) resides, it may be advantageous to move the connection to that server. Or to further have balancing of server communications load: if the total communications load on a server is substantially greater than that on some other, it may be useful to move some of the connections from the highly loaded server to the lightly loaded one, and balancing of resource access load (e.g., disk I/O load) -- as preceding but for disk I/O load rather than comm load. This is an optimization process that involves multiple dimensions, and the specific decisions made for a given set of measurements may depend on adminstrative policies, historical data about client activity, the capabilities of the various servers and network components, etc.

To this end, FIGURE 5 depicts this redistribution of client load by illustrating a connection 325 (depicted by a dotted bi-directional arrow) between client 12C and server 162. It will be understood that after redistribution of the client load, the communication path between the client 12C and server 161 may terminate.

Balancing of client load is also applicable to new connections from new clients. When a client 12F determines that it needs to access the resources provided by server group 16, it establishes an initial connection to that group. This connection will terminate at one of the servers 161, 162, or 163. Since the group appears as a single system to the client, it will not be aware of the distinction between the addresses for 161, 162, and 163, and therefore the choice of connection endpoint may be random, round robin, or fixed, but will not be responsive to the current load patterns among the servers in group 16.

When this initial client connection is received, the receiving server can at that time make a client load balancing decision. If this is done, the result may be that a more appropriate server is chosen to terminate the new connection, and the client connection is moved accordingly. The load balancing decision in this case may be based on the general level of loading at the various servers, the specific category of resource requested by the client 12F when it established the connection, historic data available to the load monitors in the server group 16 relating to previous access patterns from server 12F, policy parameters established by the administrator of server group 16, etc.

Another consideration in handling initial client connections is the distribution of the requested resource. As stated earlier, a given resource may be distributed over a proper subset of the server group. If so, it may happen that the server initially picked by client 12F for its connection serves no part of the requested resource. While it is possible to accept such a connection, it is not a particularly efficient arrangement because in that case all requests from the client, not merely a fraction of them, will require forwarding. For this reason it is useful to choose the server for the initial client connection only from among the subset of servers in server group 16 that actually serve at least some portion of the resource requested by new client 12F.

This decision can be made efficiently by the introduction of a second routing database. The routing database described earlier specifies the precise location of each separately moveable portion of the resource of interest. Copies of that routing database

need to be available at each server that terminates a client connection on which that client is requesting access to the resource in question. The connection balancing routing database simply states for a given resource as a whole which servers among those in server group 16 currently provide some portion of that resource. For example, the connection balancing routing database to describe the resource arrangement shown in Figure 1 consists of two entries: the one for resource 17 lists servers 162 and 163, and the one for resource 18 lists servers 161, 162, and 163. The servers that participate in the volumes) shown on Figure 6 will serve for this purpose. The servers that participate in the volume have the entire routing table for that volume as shown on Figure 6, while non-participating servers have only the bottom section of the depicted routing table. In the partitioning of the two volumes shown in Figure 1, this means that server 161 has the bottom left table shown in Figure 6 and both of the right side tables, while servers 161 and 162 have all four tables depicted.

At one point, the client 12C is instructed to stop making requests from server 161 and start making them to server 162. One method for this is client redirection, i.e., the client 12 is told to open a new connection to the IP address specified by the server doing the redirecting. Other mechanisms may be employed as appropriate.

Although FIG. 1 depicts the system as an assembly of functional block elements including a group of server systems, it will be apparent to one of ordinary skill in the

art that the systems of the invention may be realized as computer programs or portions of computer programs that are capable of running on the servers to thereby configure the servers as systems according to the invention. Moreover, although FIG. 1 depicts the group 16 as a local collection of servers, it will be apparent to those or ordinary skill in the art that this is only one embodiment, and that the invention may comprise a collection or group of servers that includes server that are physically remote from each other.

As discussed above, in certain embodiments, the systems of the invention may be realized as software components operating on a conventional data processing system such as a Unix workstation. In such embodiments, the system can be implemented as a C language computer program, or a computer program written in any high level language including C++, Fortran, Java or basic. General techniques for such high level programming are known, and set forth in, for example, Stephen G. Kochan, *Programming in C*, Hayden Publishing (1983).

While the invention has been disclosed in connection with the preferred embodiments shown and described in detail, various modifications and improvements thereon will become readily apparent to those skilled in the art. Accordingly, the spirit and scope of the present invention is to be limited only by the following claims.